Surrogate-Assisted Optimisation of Composite Applications in Mobile Ad hoc Networks

Dionysios Efstathiou, Peter Mcburney, Steffen Zschaler, Johann Bourcier

Surrogate-Assisted Optimisation of Composite Applications in Mobile Ad hoc Networks

Dionysios Efstathiou
Department of Informatics
King's College London, UK
name.lastname@kcl.ac.uk

Peter McBurney
Department of Informatics
King's College London, UK
peter.mcburney@kcl.ac.uk

Steffen Zschaler
Department of Informatics
King's College London, UK
szschaler@acm.org

Johann Bourcier
IRISA
University of Rennes 1, France
johann.bourcier@irisa.fr

ABSTRACT

Infrastructure-less mobile ad hoc networks enable the development of collaborative pervasive applications. Within such dynamic networks, collaboration between devices can be realised through service-orientation by abstracting device resources as services. Recently, a framework for QoS-aware service composition has been introduced which takes into account a spectrum of orchestration patterns, and enables compositions of a better QoS than traditional centralised orchestration approaches. In this paper, we focus on the automated exploration of trade-off compositions within the search space defined by this flexible composition model. For the studied problem, the evaluation of the fitness functions guiding the search process is computationally expensive because it either involves a high-fidelity simulation or actually requires calling the composite service. To overcome this limitation, we have developed efficient surrogate models for estimating the QoS metrics of a candidate solution during the search. Our experimental results show that the use of surrogates can produce solutions with good convergence and diversity properties at a much lower computational effort.

Categories and Subject Descriptors

D.2.0 [**Software Engineering**]: General

Keywords

Service composition, optimisation, surrogate models

1. INTRODUCTION

Advances in networking and hardware technology have popularised the use of powerful smart-phones and tablets, which are progressively replacing PCs for web-access [9]. These mobile devices offer a wide range of built-in sensors, fast processors, and networking capabilities. These features present great potential for creating self-configuring ecosystems of collaborating devices by forming infrastructure-less Mobile Ad hoc NETworks (MANETs) [7]. MANETs are used in various domains, such as healthcare, emergency management, and smart cities [21].

A MANET is a peer-to-peer network with no central control entity or pre-existing infrastructure. Within a MANET, data exchange is realised through multi-hop communications using intermediate nodes as relays [7]. Service Oriented Architectures (SOA) promote node-to-node collaboration in a MANET by abstracting nodes' available resources as loosely coupled software services [2]. Offered services can be called by any node participating in the network while nodes move freely and may join or leave the network at any time. One of the core ideas of SOA is the property of composability [17] where services are designed to be combined into value added structures, called service compositions. Service composition provides functionality that none of the component services could provide by themselves [24].

While the composition mechanism offers a nice programming abstraction for aggregating pieces of software, finding compositions providing the desired functionality while simultaneously exhibiting optimal Quality of Service (QoS) trade-offs is a very challenging task. In a MANET environment, nodes with heterogeneous and dynamic resources offer services with different QoS levels. There may be a large number of composite service architectures which fulfil the same functionality, but differ in their QoS such as network latency and energy consumption. Multi-Objective Evolutionary Algorithms (MOEAs) are a powerful tool for finding trade-off solutions to problems with large architectural design spaces where an exhaustive search is infeasible. The QoS metrics of interest for measuring the quality of a service composition can be used as fitness functions within a MOEA to guide the search of optimal solutions [13]. However, determining the QoS of a candidate service composition is not trivial in the case of a dynamic MANET. We can determine the QoS of a candidate composition either by simulating or actually invoking the service composition. Both approaches are computationally expensive requiring minutes or even hours for a single evaluation. This fact makes the use of such an algorithm in our problem impractical as MOEAs require a large number of fitness function evaluations before converging to a set of optimal solutions.

GECCO'14, July 12–16, Vancouver, BC, Canada.

http://dx.doi.org/10.1145/2576768.2598307 .

Current service composition optimisation approaches use synthetic fitness functions for simulating the QoS attributes of services participating in a composition [4, 26]. Others, use one-time measurements of services' QoS which are considered to remain static during the life-cycle of the composed service [22, 30]. These approaches do not take into account the computational cost for estimating the fitness functions of a composite service. On the other hand, others use a computationally expensive simulation for estimating the fitness of a solution [19] which is impractical to be integrated in the process of searching for trade-off composite services.

In this paper, we propose an efficient technique for predicting the QoS of composite applications which is crucial for supporting automated QoS-aware optimisation of composite services. Our main contributions are the following:

- We introduce low cost **surrogate models for predicting the QoS** metrics of composite services in service-based MANETs. We develop statistical models for predicting the QoS metrics of network latency, energy consumption, and success ratio.

- We present a **multi-objective surrogate-based optimisation** approach to explore trade-off composite services by **replacing the expensive** simulation-based fitness functions with **efficient surrogate models** within the evolution process of the popular NSGA-II algorithm [8]. Our experimental results show that the developed surrogates guide the NSGA-II algorithm to high quality solutions with little computational effort and by exploiting limited predictive information without the need for expensive high-fidelity simulations or real-world service invocations.

The remainder of this paper is structured as follows: In Section 2 the service composition optimisation problem is defined. Section 3 introduces our surrogate-based optimisation approach to solve this problem using the popular NSGA-II algorithm. Section 4 presents the considered use-case scenario and the research questions we seek to answer. Section 5 presents the experimental results of our study followed by Section 6 which discusses related work. Finally, Section 7 presents conclusions and future work.

2. MOTIVATION AND PROBLEM

In this section, we first describe the motivating scenario of our study and then we formulate the search problem of composing services in a distributed service-based environment.

2.1 Motivating Scenario

Due to their distributed and infrastructure-less nature, multi-hop ad hoc wireless networks are ideal for building crisis management applications [6]. Our motivating scenario considers a decision support system for improving the decision making of fire-fighters in an emergency situation such as a forest fire. In the considered scenario, fire-fighters are equipped with mobile devices which form a MANET. These devices, following SOA principles, offer their resources (e.g. application data, network and hardware components) as loosely coupled software services. Service composition promotes the collaboration between devices by composing services provided by various devices towards achieving complex applications.

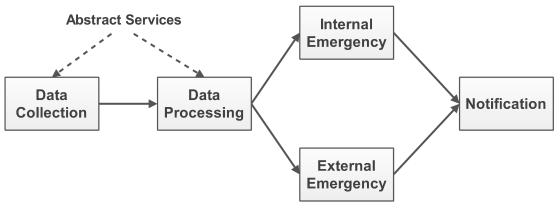

Figure 1: An example service composition plan.

For example, consider the scenario where a commanding firefighter uses a composite application comprised of five services to infer whether a firefighter is in danger and appropriate measures should be taken in time. More specifically, the commanding firefighter aggregates information about the condition of his subordinates (position, heart rate, oxygen level). This information needs to be fed to a processing service which assesses if something is going wrong. In the case of an emergency event (e.g. a firefighter has stopped moving and high levels of carbon dioxide in the blood are observed), another team in the close proximity must be notified to intervene along with a medical team which must be ready to approach. Finally, the various decisions and the situation event are logged to a central database which may respond with a set of recommended actions to be taken.

In such a dynamic and time-critical scenario where resources are limited, composite applications should exhibit QoS properties such as minimum response time and battery consumption. However, the QoS of an offered service is influenced by a variety of factors such as topology changes, user's mobility, obstacles, resource availability (e.g. battery level), and others. QoS of a composite application are highly susceptible to changes on the underlying network.

Our goal is to provide to the users of the distributed system with composite services which exhibit optimal QoS trade-offs at a small computational overhead. It is worth noting that our approach is not restricted to the presented scenario but it can be applied to others involving mobile multi-hop communication networks. Such applications can be found in the domain of *Mobile Enterprise Vision* [16].

2.2 Problem Formulation

Our problem formulation starts from the abstract plan of a composite application which describes the services to participate in the composition. Within the considered network, various nodes offer concrete services which implement the abstract services of the plan. Also, orchestrator nodes are responsible for calling services and forwarding the intermediate results to the appropriate nodes. Our goal is to produce optimal composition configurations by tuning the parameters of a service composition configuration, called Degrees of Freedom (DoF) which are the following [10]: (a) selection of concrete services, (b) partitioning of the composition into sub-orchestrations, and (c) selection of orchestrating nodes.

Def. 1. Distributed Service Orchestration Problem Given: A set of m abstract services that compose the service composition plan P, a set of n nodes participating in the network where each node provides a single concrete service and can coordinate a single orchestration, a mapping of the n available concrete services that implement the functionality of the m abstract services, and a set of q quality objectives $Q = \{Q_1, \cdots, Q_q\}$.

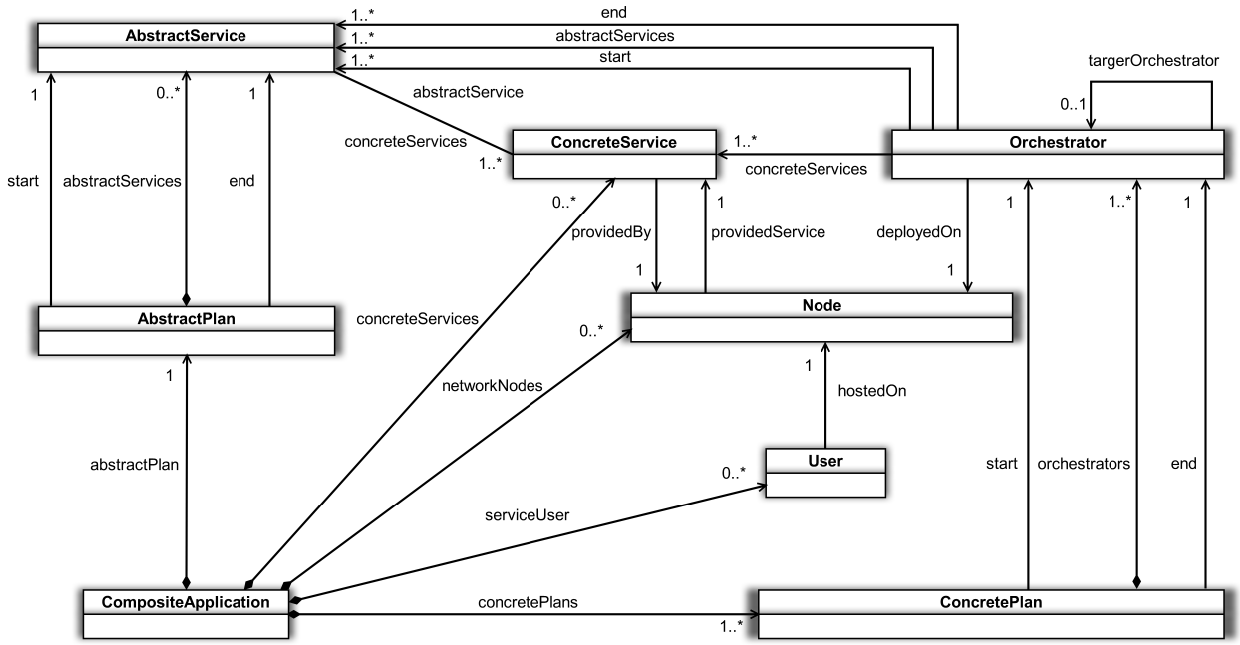

Figure 2: The composition configuration metamodel.

Problem: Find a set of service composition configurations, all of which implement the functionality described by P, but differ in their non-functional trade-offs according to Q.

Let a composite application plan P be represented as a directed graph, as shown in Fig. 1, consisting of a node set $AS = \langle AS_1, AS_1, \cdots, AS_n \rangle$ of abstract services and an edge set $DF = \langle (AS_i, AS_j) : i \neq j,\ 1 \leq i \leq n, 1 \leq j \leq n \rangle$ of data flow between abstract services, where AS_i is the source and AS_j the data destination.

2.3 QoS Metrics - Fitness Functions

QoS metrics of software architectures can be used as fitness functions to guide the search for optimal solutions for Search-Based Software Engineering (SBSE) problems [13]. In our study, we consider the following QoS metrics for estimating the quality of a composite service:

- *The network latency* Q_{NL} which measures the round trip time of the exchanged messages within a service composition. This metric is computed based on the network latency of successfully delivered messages.

- *The energy consumption* Q_E of a configuration, which is the energy difference observed in the nodes for realising a service composition configuration where nodes spent energy for: (a) sending/receiving data, and (b) orchestrating other services. They also spend energy for service execution, but this is independent of the configuration; therefore we can safely ignore it.

- *The success ratio* Q_{SR} is calculated as the percentage of successfully exchanged data between collaborating nodes within a composition configuration.

3. THE PROPOSED APPROACH

In this section, we describe our solution approach to the expensive optimisation service composition problem. We propose the adoption of a Search-Based Software Engineering (SBSE) [14] approach for exploring service composition configurations that exhibit quality trade-offs according to the three QoS metrics described in Section 2.3. To build a search-based approach we need to define the following two ingredients [14]: (i) a problem representation, and (ii) the appropriate fitness functions. We propose the replacement of the computationally expensive "real" simulation-based fitness functions with efficient surrogate models. The last step is to employ a search algorithm that utilises the defined fitness functions for exploring the space of candidate solutions.

3.1 Defining the Design Space

We first define the space of service composition configurations. The work in [10] showed that by taking into account multiple DoFs for tuning the quality of a composite service, a space containing solutions of higher quality than the centralised orchestration which considers only the DoF of service selection can be defined. More specifically, the authors proposed the following parameters for tuning the quality of a service composition configuration are the following: (a) service selection, (b) orchestration partitioning, and (c) orchestrator node selection. The large number of parameters affecting the performance of a configuration and their interdependencies, motivate the engineering of an automated method for design space exploration. We call a composition configuration a *solution*. The set of all candidate solutions is the set of all possible combinations of the defined DoFs.

Fig. 2 depicts our metamodel for specifying composition configurations in distributed service-based environments. The proposed metamodel abstracts away from the complex-

ity of the underlying system and focuses only on the concepts related to the problem of distributed composition.

The *Composite Application* meta-class represents a high-level application to be realised as requested by a *User*. This high-level application is described as an *Abstract Plan* which combines a number of *Abstract Services*. An instance of *Abstract Plan* would be, for example, an abstract composition workflow such as the one depicted in Fig. 1. The connection order of the various *Abstract Services* describes the order of invocation (control flow) among the various participating services in the composition. An *Abstract Service* describes in an abstract way the functionality to be fulfilled by a concrete invocable realisation of a service which is called *ConcreteService*. As shown in our metamodel, each *Abstract Service* can be implemented by many *Concrete Services* which can be differentiated based on their provided QoS. For example, two *Concrete Services* provide the same functionality, in other words implement the same *Abstract Service*, but one may be faster than the other.

To enable the partitioning of the initial abstract workflow to sub-orchestrations, we introduce the concept of the *Orchestrator* which is responsible for composing a set of *Concrete Services* by controlling their order of execution and data flow. The *Nodes* meta-class represents a physical node that participates in the system and provides a *Concrete Service*. Another important role of a *Node* is its ability to provide an *Orchestrator*. Based on the resource-constrained nature of the studied scenario, we assume that a node can provide a single *Concrete Service* and host an *Orchestrator* at a time.

An instance of meta-class *Concrete Plan* describes a concrete realisation plan for a *Composite Application* which may be realised by more than one *Concrete Plans*. In other words, there can be many possible configurations for realising the same application which share functionality but which may differ in their provided QoS.

3.2 Computational Search

We employed the popular NSGA-II algorithm for exploring trade-off composite services which is based on two core ideas of Pareto ranking for exploiting the best solutions and crowding distance for promoting diversity within a population. To apply the NSGA-II algorithm in our problem, we have to define the following building blocks: (a) solution representation, (b) genetic operators, and (c) fitness functions, which are described in the subsections below.

3.2.1 Solution Representation

The service composition configuration chromosome constitutes a container for further chromosomes that correspond to classes shown in our metamodel (Fig. 2). The genotypes exhibit variable lengths due to the fact that the number of sub-orchestrations can be varied between one (single centralised orchestrator) and m (fully decentralised orchestration where one orchestrator is responsible for orchestrating a single abstract service).

We represent a candidate solution to our problem as a three-fold assignment of (i) concrete services to abstract services, (ii) abstract services to sub-orchestrations and (iii) network nodes to orchestrator nodes. Fig. 3(a) shows the chromosome for mapping concrete services to abstract services. The chromosome for representing a sub-orchestration is depicted in Fig. 3(b). Finally, a complete configuration can be represented as a set of sub-orchestration chromosomes as denoted by the + sign in the upper left corner of the box shown in Fig. 3(c). The grey box indicates the chromosome, while the white boxes represent the genes that compose the chromosome. The signs 1, + and ? represent the repetition of a gene or chromosome exactly once, at least once, and at most once respectively.

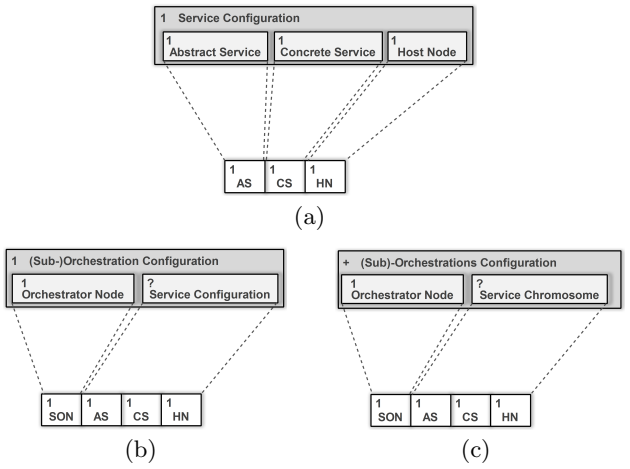

Figure 3: Chromosome representations for: (a) mapping concrete to abstract services, (b) sub-orchestrations, and (c) composition configuration.

An example chromosome representation for a complete composition configuration is depicted in Fig. 4. In this example, we have two sub-orchestrations which are hosted on nodes (SON) 2 and 4 respectively. The first sub-orchestration includes the abstract services 1 and 2, while the second orchestrates the services 3, 4 and 5. For implementing the abstract service 1 (AS), the concrete service 12 (CS) was selected which is hosted at node 2 (HN).

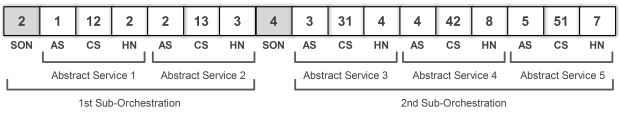

Figure 4: Example solution chromosome for a composition configuration with 2 sub-orchestrations.

3.2.2 Genetic Operators

The design of genetic operators is the second key element for constructing any MOEA. The chosen varied size representation complicates the implementation of genetic operators. For each of the three considered DoFs, we designed a pair of crossover and mutation operators, resulting to a total of 6 operators. *Crossover* ensures that features of the parents will be passed to the offspring forcing the convergence on the good solutions found so far (*exploitation*). For example, the single-point crossover for the first DoF, produces a new configuration by combining the concrete services mappings of two parents as depicted in Fig. 5.

For the second DoF, two parent configurations are combined by swapping sub-orchestrations between them. The crossover operator for the last DoF, creates an offspring by combining the list of orchestrators of the parent configura-

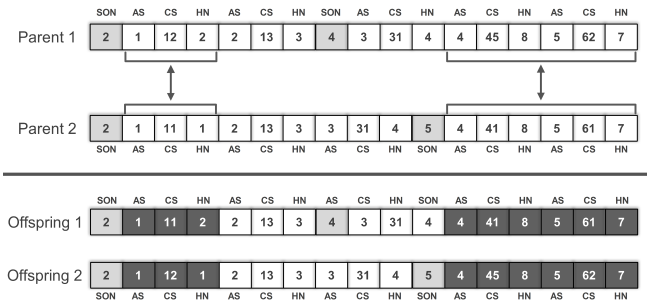

Figure 5: An example crossover operation for the first DoF.

tions. Moreover, *mutation* ensures that any possible configuration can be searched which increases the *exploration* of the space and aims at maintaining diversity by avoiding to over-bias to fittest individuals. For instance, the mutation operator for the first DoF chooses randomly a concrete service for implementing an abstract service. All the designed operators allow the modification of solutions to reach any area of the space while guaranteeing that the changed configuration is valid and conforms to our described metamodel.

3.2.3 Fitness Function

Fitness function is a quality measure of a candidate solution and guides the search process by distinguishing between better and worse solutions. For estimating the QoS metrics described in Section 2.3, we can either employ a computationally expensive experiment or simulation, or light-weight surrogate models to approximate them.

Expensive Function: As we do not have the resources for performing real-world experiments of the studied scenario, we replaced them by a less costly simulation. However, computing the QoS metrics of a candidate composition configuration by using the simulation ($\sim$ 300 seconds) is not fast enough for integrating it in an optimisation approach.

Surrogate Function: MOEAs are efficient at exploring a large and complex search space by providing several Pareto solutions in a single run at the cost of a large number of fitness function evaluations. MOEAs seem unaffordable in our problem which involves expensive fitness functions because their computational cost has a critical impact on the overall complexity of the search algorithm [13].

We have two choices for reducing the computation cost of a MOEA [18]: (a) reduce its algorithmic complexity, or (b) reduce the computational cost of the fitness function. The first approach does not seem promising because the main overhead when applying a MOEA in a real-world problem is because of the complex nature of the tackled problem. For the second approach, functional approximation [18] proposes the replacement of the expensive fitness functions with cheaper alternatives called surrogates [18, 23]. Surrogates are statistical models that are built by exploiting knowledge of already evaluated solutions (training dataset) of the real simulation or experiment-based fitness functions [19, 20]. The surrogates can be used by the MOEA to evolve the population of individuals at a smaller computational cost than that of the expensive real fitness function.

By constructing a surrogate model our goal is to predict the QoS of a composite service where service users and providers communicate over a wireless MANET. First, we need to determine the factors which may affect the QoS metrics of a composite service described in Section 2.3.

Symbol	Description
PV_1	# of hops between nodes in a composition
PV_2	# of transmitted packets in a composition
PV_3	# of hops of the longest path in a composition
PV_4	# of orchestrator nodes in a composition
PV_5	# of neighbours of each node in a composition
PV_6	# of routes between nodes in a composition
PV_7	Total distance between nodes in a composition

Table 1: Variables for predicting the QoS of composite applications.

Regression analysis [15] focuses on developing models for describing how a response variable is related to one or more predictor variables. When developing a prediction model, each predictor variable should be able to provide some information about the value of the response not already provided by any other variable.In our study, we consider the seven predictor variables ($p = 7$) shown in Table 1. These predictors are not the only ones which we could use for our study but we have chosen them for our analysis because it was comparatively easy to obtain the relevant data for them. In the last column of Table 1 we show the cost of acquiring each predictor variable. Also, we use the following approaches for creating surrogate models: (1) Linear Regression models (LR) [15], (2) Multivariate adaptive regression splines (MARS) [15], (3) Classification and Regression Trees (CART) [15], and (4) Random Forests (RF) [15].

In [11], you can find more details about how we developed the used surrogate models and more specifically the selection of their predictor variables and parameters.

4. EMPIRICAL STUDY DESIGN

We now describe our experimental case study, the metrics for the algorithms in comparison and our research questions.

4.1 Firefighting Case Study

We use a simulation-based approach for collecting our dataset for building the used surrogate models. We use the Network Simulator 3 (NS-3) [1] to simulate a service-based firefighter decision support system where firefighters of three different hierarchical levels (Group, Engine, and Team) carry devices which offer software services and cooperatively form an infrastructure-less MANET. In detail, we simulate a network of 63 mobile nodes (3 Group Leaders, 12 Engine Leaders, and 48 Team Leaders/Members) with transmission range of $45m$ distributed in a area of $500m \times 500m$. Each group follows a different mobility model because each group has a different purpose and mission to fulfil. To ensure that the network is completely configured before simulating a composition configuration we included a set-up/warm-up time of 20 seconds. For more details about the simulation scenario such as the chosen mobility, network and routing parameters, please refer to our previous work in [10].

4.2 Experimental Setting

Our prototype used the MOEA Framework[2] and the statistical environment R [25] for the various surrogate

[1] `http://www.nsnam.org/`

[2] `http://www.moeaframework.org/`

techniques. We took advantage of the elasticity provided by Amazon Elastic Compute Cloud (EC2) for executing the computationally-expensive experiments including the simulation-based fitness functions. The NSGA-II algorithm evolved a population of 96 configurations for 30 generations. After some tuning, we resulted with the following parameters for our problem: 95% crossover rate, 5% mutation rate.

4.3 Research Questions

Our experimental study was conducted to assess the applicability of surrogate models in the multi-objective service composition optimisation problem and to compare the efficiency of state of the art approaches. More specifically, we aimed to answer the following research questions:

RQ1 How does NSGA-II perform compared to random search for the studied problem?

RQ2 Can the developed cheap surrogate models replace the expensive simulation-based functions within the EA towards exploring good optimal solutions?

RQ3 What is the performance gain and loss of optimality by using a light-weight surrogate model instead of using the real expensive fitness function in our problem?

4.4 Evaluation Metrics Used

To quantitatively compare the performance of NSGA-II for the cases of the expensive and the efficient surrogates we employ three well-known quality indicators, namely Hypervolume (I_{HV}) [31], Cardinality (I_C) [31], and Spread (Δ) [8]. To compute them, we normalise fitness values and use as a reference a nadir point containing the worst possible QoS metrics for a composition configuration.

I_C of a Pareto set approximation measures the number of optimal solutions either in decision or objective space. Cardinality is very cheap to compute, but the indicator is not monotonic, or in other words, it is not compatible with the Pareto dominance relation. I_{HV} calculates the volume of the objective space which is weakly dominated by a specific Pareto set approximation. The higher the hypervolume, the better the quality of the approximation set. I_{HV} is currently the only unary indicator known to be strictly monotonic [31]. However, this advantage comes at the cost of high computational overhead which is exponential in the number of objectives. Δ is a diversity metric that measures how evenly the points in the approximation set are distributed in the objective space. A smaller spacing value indicates that the population has a better more uniform spread.

4.5 Statistical Analysis

In stochastic optimisation the relationship between quality of the resulted solutions and necessary computational resources is not fixed, but can be described by a probability density function making every statement about their performance probabilistic in nature. If we apply the same algorithm several times to the same problem, each time a different Pareto approximation may be returned. To obtain reliable conclusions about the performance of the inherently stochastic MOEAs and the quality of the generated solutions, we are forced to use statistical tools. In our experiments, we repeat each algorithm 20 times and collect the values of the corresponding quality/performance indicators.

	I_{HV}		I_C		Δ	
	Mean	σ	**Mean**	σ	**Mean**	σ
Random	0.503	$\pm$ 0.03	13.1	$\pm$ 4.82	0.8128	$\pm$ 0.12
NSGA-II	0.6669	$\pm$ 0.01	30.9	$\pm$ 5.5	0.6954	$\pm$ 0.09

Table 2: Hypervolume (I_{HV}), Cardinality (I_C), and Spread (Δ) of random search and NSGA-II with the expensive fitness functions.

A statistical test is used to asses whether the observed differences between compared algorithms are statistically significant [1]. We use a non-parametric Mann-Whitney test to evaluate statistical significance because we have no information about the distribution of the data.

5. RESULTS AND DISCUSSIONS

Results for RQ1: To answer **RQ1** we implemented a random search algorithm which assigns random values for each of the three degrees of freedom of a candidate composite service. More specifically, we run both random search and NSGA-II by using the simulation-based expensive objective function and report the quality indicators described in Section 4.4. Finally, we compare the Pareto surfaces produced by NSGAII for both composition models in comparison.

To answer **RQ1**, we compared the quality of the Pareto solutions discovered by random search and NSGA-II. The null hypothesis (H_0) states that the approaches using a dummy random search and NSGA-II produce solutions of the same quality. H_0 is rejected by the Mann-Whitney test at 1% significance level (p-values for the three indicators $< 3.4 \cdot 10^{-8}$, $< 6 \cdot 10^{-8}$, and $< 3 \cdot 10^{-3}$ respectively).

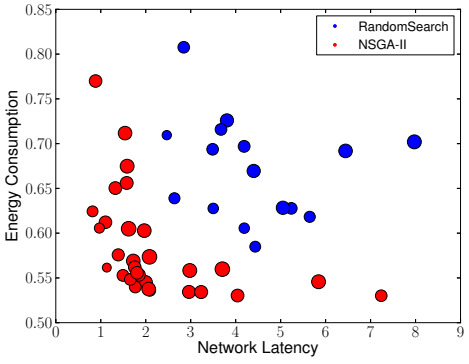

Figure 6: Pareto fronts where the size of the points represents the Success Ratio QoS dimension (the larger the point size, the higher the success ratio).

Table 2 presents the mean and standard deviation for the three quality indicators described in Section 4.4. According to the results, the quality indicators achieved by NSGA-II are significantly better than those of random search. To qualitatively support our point, we present in Fig. 6 a visualisation of the Pareto fronts achieved by the two approaches in comparison. Based on that figure we observe that the NSGA-II algorithm is able of achieving higher quality, more and better spread Pareto optimal solutions.

	I_{HV}		I_C		Δ		$t_{Execution}$	$\frac{HV_{Surrogate}}{HV_{Expensive}}$	$\frac{t_{Expensive}}{t_{Surrogate}}$
	Mean	σ	**Mean**	σ	**Mean**	σ	**(seconds)**		
Expensive	0.6669	± 0.01	30.9	± 5.5	0.6954	± 0.09	13376.5	-	-
LR	0.6027	± 0.21	45.7	± 21.27	0.6747	± 0.06	1.4644	0.9037	9134
MARS	0.6414	± 0.02	15.9	± 4.78	0.8585	± 0.13	1.8808	0.9618	7112
CART	0.6161	± 0.02	9.4	± 1.91	0.771	± 0.09	1.9593	0.9238	6827
RF	0.6095	± 0.02	20.7	± 12.43	0.7752	± 0.14	3.0225	0.9139	4425

Table 3: Quality indicator values of simulation-based and surrogate fitness functions.

Results for RQ2: When evaluating surrogate models in the context of evolutionary computation we need to assess the ability of the developed surrogates to guide the search towards good areas of the service composition space. With regards to **RQ2**, we first run the NSGA-II algorithm with the simulation-based fitness function. Then, we replace the expensive fitness functions with the surrogate models and compute the simulation-based QoS metric values on the Pareto individuals of the last generation. In the first three columns of Table 3, we present the results for the three indicators obtained after 20 runs.

Table 3 shows that the use of the expensive fitness functions achieves the best hypervolume results as expected. Among the surrogate models, MARS achieves the highest I_{HV}. For the I_C and spread (Δ) indicators, LR achieves the best results due to its ability to assign different QoS values by doing fine-grain predictions based on slightly different predictor values. Tree-based approaches, like CART and RF, seem unable to provide fine-grain differentiation of solutions. These approaches try to handle the trade-off between building a big tree with many leaves for describing accurately a specific training set (overfitting) and generalisation of performance. LR results in Pareto fronts with many solutions but not of very good quality. The H_0 for **RQ2** states that the approaches using the expensive and surrogate functions produce Pareto solutions of the same quality is rejected by the Mann-Whitney test at 1% significance level but due to space restrictions we do not report the p-values calculated for each pair of the compared techniques.

Results for RQ3: We now study the trade-off between the accurate but expensive simulator-based fitness functions with the less accurate but computationally cheap surrogate models. To evaluate the efficiency of the various surrogate models, we present in the last three columns of Table 3: (a) the execution time ($t_{execution}$) of the various fitness functions needed to evaluate the QoS of a population of solutions within a generation of NSGA-II, (b) the quality degradation based on the hypervolume indicator of the approach using the surrogate models in relation to the best performance achieved by the accurate simulation-based fitness functions, and (c) the execution speed-up achieved by the surrogate-based approach over the one using the expensive fitness functions. Note that the one-time upfront cost of building the model is negligible and it is not reported (< 5 seconds for all the models). All the experiments were executed in R using a machine with Intel Core i7 vPro with 12GB DDR3 RAM, running the Linux 3.2.0-40 kernel.

We seek an appropriate balance between the accuracy of the various surrogate models and their estimation cost. It is obvious that, due to its simplicity *LR* achieves the largest speed-up but also the worst results in terms of hypervolume. Also, MARS appears to be the best performing technique and its speed-up is comparable to the simple *LR* technique. As expected *RF* is more time-consuming than CART as it involves the construction of multiple decision trees to form an ensemble forest. It is worth noting that all of the reported model build times are small compared to the time needed for running the expensive fitness function for a single service composition configuration (~ 3 *minutes*) in the context of our studied motivating scenario.

6. RELATED WORK

The work related to our approach can be classified into two groups. The first group concerns optimising service composition by selecting the optimal set of concrete services to participate in a centralised orchestration. To achieve this, existing approaches use different techniques such as heuristics [22], genetic algorithms [3, 4], linear programming [5, 29], swarm intelligence [28], and others. However, their main limitation is that they neglect the inherent problems of centralised orchestration and try to optimise the QoS of a composite service by choosing in isolation which services to participate without considering how these services are composed together. Also, they assume that the services' QoS is known beforehand or is artificially generated without taking into account the cost of fitness function estimation.

The second group focuses on how to predict the QoS of single and composite services. Gambi *et al.* [12] proposed the use of surrogate models for building SLA protection controllers for single RESTful services. Zheng *et al.* [30] proposed a collaborative filtering approach for predicting QoS values of Web services based on past experiences of service users. Wang *et al.* [27] proposed a prediction-based approach to maintain the QoS of a composite service during execution. The authors study the service selection process and propose a probabilistic approach for modelling the reliability of services and service compositions. However, these approaches are focusing on predicting the QoS of composite services based on past QoS of the component services ignoring how these services are combined together. Also, they are based on the assumption that past QoS measurements remain static during the life-cycle of the composed service which is unrealistic for the considered dynamic environments. Finally, they focus on traditional web services offered via wired and resource-rich networks.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a surrogate-assisted multi-objective optimisation approach for exploring optimal service compositions in the context of service-based MANETs. To the best of our knowledge, this is the first work to introduce the use of approximation models as fitness functions for

exploring optimal service composition configurations. We explored four well-known surrogate models for predicting the QoS of service compositions which we integrated to the popular NSGA-II algorithm. The experimental results provided evidence that efficient surrogate models can guide the evolutionary search into exploring solutions of high quality.

As future work, we plan to include in our research the response time metric which depends on the processing speed and current load of the devices and the various services. For that reason, we need to include a more detailed model of the services which run on the devices of the scenario and their response time based on various processing loads.

Acknowledgements. This work has been supported by the European FP7 Marie Curie Initial Training Network "RELATE" (Grant Agreement No. 264840).

8. REFERENCES

[1] A. Arcuri and L. C. Briand. A Practical Guide for Using Statistical Tests to Assess Randomized Algorithms in Software Engineering. In *ICSE*, 2011.

[2] J. Bourcier and C. Escoffier. Implementing Home-Control Applications on Service Platform. In *4th CCNC*, 2007.

[3] M. Bozkurt. Cost-aware Pareto Optimal Test Suite Minimisation for Service-centric Systems. In *GECCO*, pages 1429–1436, 2013.

[4] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani. An Approach for QoS-Aware Service Composition Based on Genetic Algorithms. In *GECCO*, 2005.

[5] V. Cardellini, E. Casalicchio, V. Grassi, S. Iannucci, F. L. Presti, and R. Mirandola. MOSES: A Framework for QoS Driven Runtime Adaptation of Service-Oriented Systems. *TSE*, 99, 2011.

[6] T. Catarci, M. de Leoni, A. Marrella, M. Mecella, B. Salvatore, G. Vetere, S. Dustdar, L. Juszczyk, A. Manzoor, and H. L. Truong. Pervasive Software Environments for Supporting Disaster Responses. *IEEE Int. Comp.*, 12(1):26–37, 2008.

[7] S. Corson and J. Macker. Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations, 1999.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II. *IEEE Trans. Evol. Comp.*, 6:182–197, 2000.

[9] T. M. T. Do, J. Blom, and D. Gatica-Perez. Smartphone Usage in the Wild: A Large-Scale Analysis of Applications and Context. In *ICMI*, pages 353–360, 2011.

[10] D. Efstathiou, P. McBurney, S. Zschaler, and J. Bourcier. Flexible QoS-Aware Service Composition in Highly Heterogeneous and Dynamic Service-Based Systems. In *WiMob*, 2013.

[11] D. Efstathiou, P. McBurney, S. Zschaler, and J. Bourcier. Efficiently Approximating the QoS of Composite Services in Mobile Ad-Hoc Networks. Technical Report TR-14-01, King's College London, January 2014. `http://www.dcs.kcl.ac.uk/staff/eustathi/papers/efstathiouTR-14-01.pdf`.

[12] A. Gambi, G. Toffetti, and M. Pezzè. Protecting SLAs with Surrogate Models. In *PESOS*, pages 71–77. ACM, 2010.

[13] M. Harman and J. Clark. Metrics Are Fitness Functions Too. In *METRICS*, 2004.

[14] M. Harman, P. McMinn, J. T. Souza, and S. Yoo. Search Based Software Engineering: Techniques, Taxonomy, Tutorial. In *Empirical Software Engineering and Verification*, volume 7007, pages 1–59. 2012.

[15] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2003.

[16] HP. Enterprise Mobility Leveraging the Mobility Economy in the Enterprise. White paper, 2012.

[17] M. N. Huhns and M. P. Singh. Service-Oriented Computing: Key Concepts and Principles. *IEEE Internet Computing*, pages 75–81, 2005.

[18] Y. Jin. A Comprehensive Survey of Fitness Approximation in Evolutionary Computation. *Soft Computing*, 9(1):3–12, 2005.

[19] Y. Jin. Surrogate-Assisted Evolutionary Computation: Recent Advances and Future Challenges. *SEC*, 1(2):61–70, 2011.

[20] Y. Jin and J. Branke. Evolutionary Optimization in Uncertain Environment - A Survey. *IEEE Trans. Evol. Comput.*, 9(3):303–317, 2005.

[21] libelium. 50 Sensor Applications for a Smarter World, 2013.

[22] N. B. Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, and V. Issarny. QoS-Aware Service Composition in Dynamic Service Oriented Environments. In *Middleware*, 2009.

[23] Y. S. Ong, P. Nair, A. Keane, and K.-W. Wong. Surrogate-Assisted Evolutionary Optimization Frameworks for High-Fidelity Engineering Design Problems. In *Knowledge Incorporation in Evolutionary Computation*, pages 307–332. 2004.

[24] M. Papazoglou and D. Georgakopoulos. Introduction: Service-Oriented Computing. *Comm. of ACM*, 46:24–28, 2003.

[25] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2013.

[26] F. Rosenberg, M. Muller, P. Leitner, A. Michlmayr, A. Bouguettaya, and S. Dustdar. Metaheuristic Optimization of Large-Scale QoS-aware Service Compositions. In *SCC*, pages 97–104, 2010.

[27] H. Wang, H. Sun, and Q. Yu. Reliable Service Composition via Automatic QoS Prediction. In *SCC*, 2013.

[28] P. Wang, K.-M. Chao, and C.-C. Lo. On Optimal Decision for QoS-Aware Composite Service Selection. *Expert Systems with Applications*, 37(1):440–449, 2010.

[29] L. Zeng, B. Benatallah, A. H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-Aware Middleware for Web Services Composition. *IEEE Trans. Soft. Eng.*, 30:311 – 327, 2004.

[30] Z. Zheng, H. Ma, M. Lyu, and I. King. QoS-Aware Web Service Recommendation by Collaborative Filtering. *IEEE Trans. Serv. Comp.*, 4(2):140–152, 2011.

[31] E. Zitzler, J. Knowles, and L. Thiele. Quality Assessment of Pareto Set Approximations. In *Multiobjective Optimization*, pages 373–404. 2008.